1、@class的使用

作用

可以简单地引用一个类

简单使用

@class Dog; //类的引入

仅仅是告诉编译器：Dog是一个类；并不会包含Dog这个类的所有内容

具体使用

在.h文件中使用@class引用一个类

在.m文件中使用#import包含这个类的.h文件

```
#import <Foundation/Foundat
@class ren;
@interface che : NSObject
{
    ren* r;
}
@end
```

1.作用

好处:如果xxxx的文件内容发生改变,引入xxxx的文件不需要改变.

特殊用法:

可以解决循环引入问题,当两个类相互引入时,会导致循环引入的出现,会出现报错,无法进行编译,

```
#import <Foundation/Foundat
@class ren;
@interface che : NSObject
{
    ren* r;
}
@end
```

```
#import <Foundation/Foundatio
@class ren;
@interface che : NSObject
{
    ren* r;
}
@end
```

```
#import <Foundation/Foundat
@class ren;
@interface che : NSObject
{
    ren* r;
}
@end
```

```
#import <Foundation/Foundation.h>
//#import 作用:
// 把要引用的头文件的内容,拷贝到写 #import处
// 如果 Person.h文件件内容发生变化了,此时所有引用 Person.h这个头文件的类,都需要重新编译
#import "Person.h"
```

- 问题:
  1，@class的作用是什么？
  2，（面试题）#import和@class的区别

2.区别

这两种的方式的区别在于：

1）#import方式会包含被引用类的所有信息，包括被引用类的变量和方法；@class方式只是告诉编译器在A.h文件中 B *b 只是类的声明，具体这个类里有什么信息，这里不需要知道，等实现文件中真正要用到时，才会真正去查看B类中信息；

2）使用@class方式由于只需要只要被引用类（B类）的名称就可以了，而在实现类由于要用到被引用类中的实体变量和方法，所以需要使用#import来包含被引用类的头文件；

3）通过上面2点也很容易知道在编译效率上，如果有上百个头文件都#import了同一 个文件，或者这些文件依次被#improt（A->B, B->C,C->D…），一旦最开始的头文件稍有改动，后面引用到这个文件的所有类都需要重新编译一遍，这样的效率也是可想而知的，而相对来 讲，使用@class方式就不会出现这种问题了；

所以：我们实际开发中尽量在.h头文件中使用@class

面试题#import和@class的区别。

作用上的区别

#import会包含引用类的所有信息(内容)，包括引用类的变量和方法

@class仅仅是告诉编译器有这么一个类，具体这个类里有什么信息，完全不知

效率上的区别

如果有上百个头文件都#import了同一个文件，或者这些文件依次被#import,那么一旦最开始的头文件稍有改动，后面引用到这个文件的所有类都需要重新编译一遍 ，编译效率非常低

14-【理解】循环retain问题

本小节知识点：

1、【理解】循环retain的问题

1，如何解决对象间属性循环引用(retain)?

```objc
#import <Foundation/Foundation.h>
#import "Person.h"
@interface Dog : NSObject
@property (nonatomic,retain) Person *owner;
@end
```

```objc
#import <Foundation/Foundation.h>
#import "Dog.h"

@interface Person : NSObject
//人拥有一条狗
@property (nonatomic,retain) Dog *dog;
@end
```

```objc
Person *p = [Person new];  //1
Dog *d = [Dog new];   //1

//人有一条狗
p.dog = d;    //  d   2
d.owner = p;  //  p   2

[p release];  //0
[d release];  //
```

一旦赋值以后,后面就会造成内存泄露

循环retain需要两个相互包含的对象都进行赋值后才能实现

如何解决?

1.再释放时让其中一个多释放一次(注意释放顺序)

2.让其中一个assgin,一个使用retain

注意用assgin的对象的delloc对象中,需要把另一个对象的release给取消掉!